```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.control.Label;
import javafx.scene.control.Button;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Polygon;

public class JavaFXApp extends Application {

    @Override
    public void start(Stage window) {
        System.out.println("app started...");

        Label label = new Label("This is a label...");
        Button button = new Button("Click Me!");
        button.setLayoutY(25);

        Circle circle = new Circle(200, 200, 20);
        Polygon polygon = new Polygon(100, 150, 180, 90, 35, 80);
```

(x,y)

Triangle here

```java
        Group group = new Group(label, button, circle, polygon);
```

container

```java
        Scene scene = new Scene(group, 800, 600);
        window.setScene(scene);
```

create an UI with 800x600

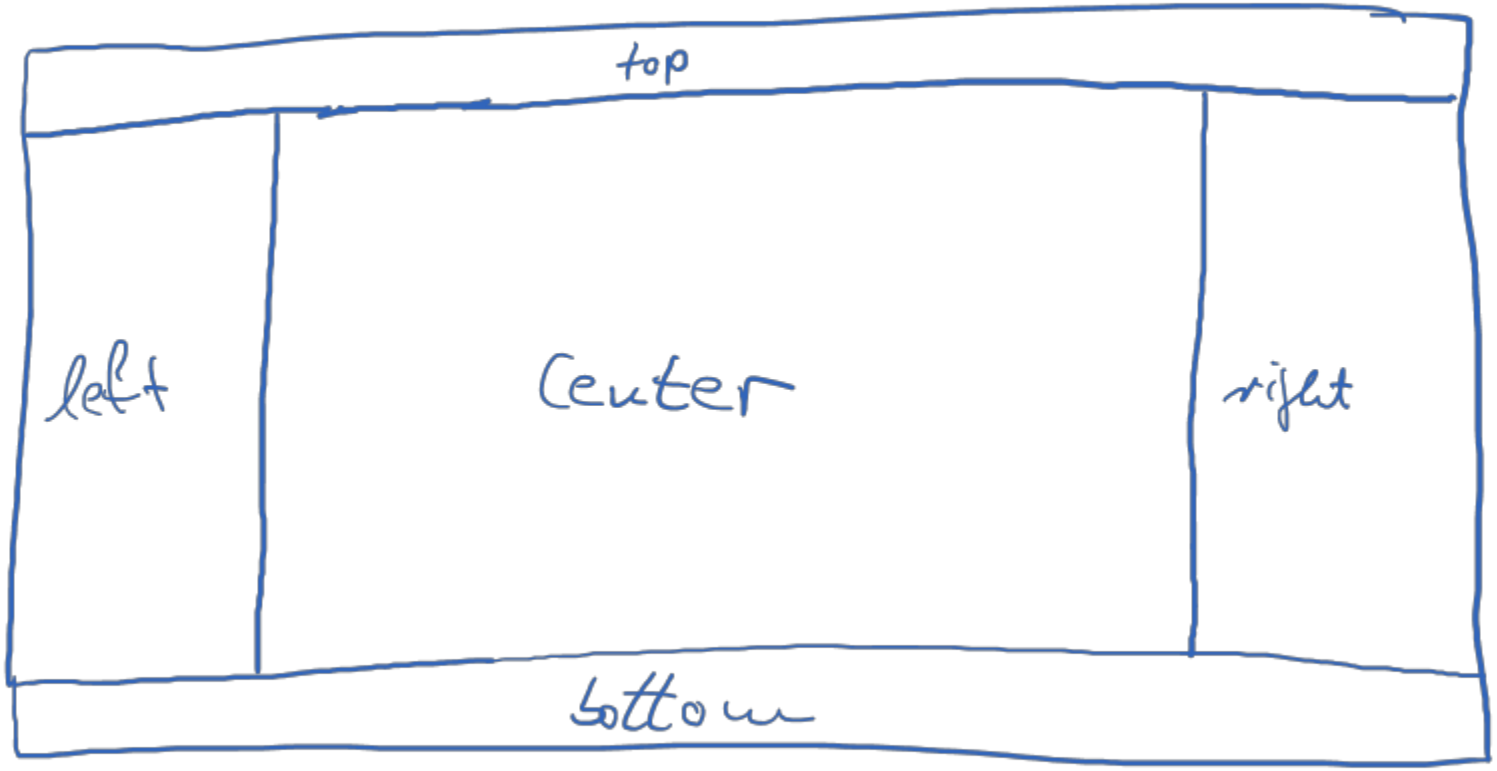
```java
        window.setTitle("JavaFXApp");
        window.show();
    }

    public static void main(String[] args) {
        Application.launch();
    }

}
```

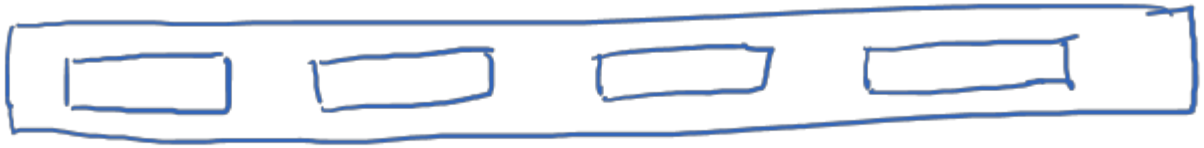
JavaFX Layout Managers

Layout Manager: BorderPane

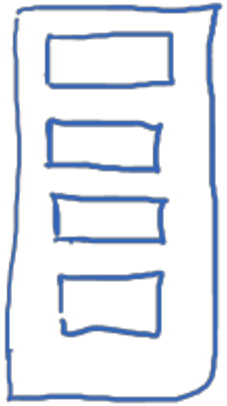


Layout Managers: HBox and VBox

- HBox : horizontal row

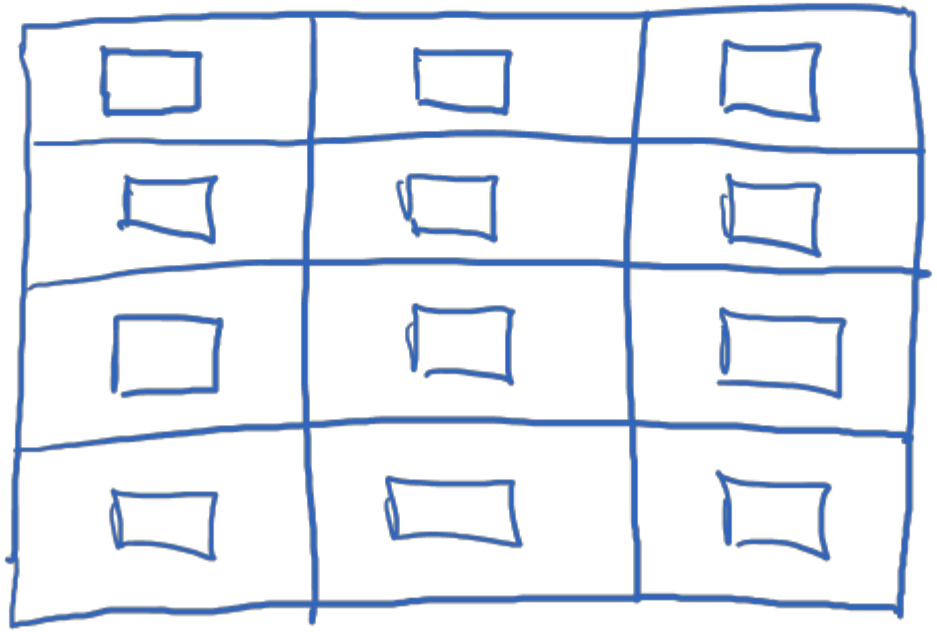


- VBox : vertical column

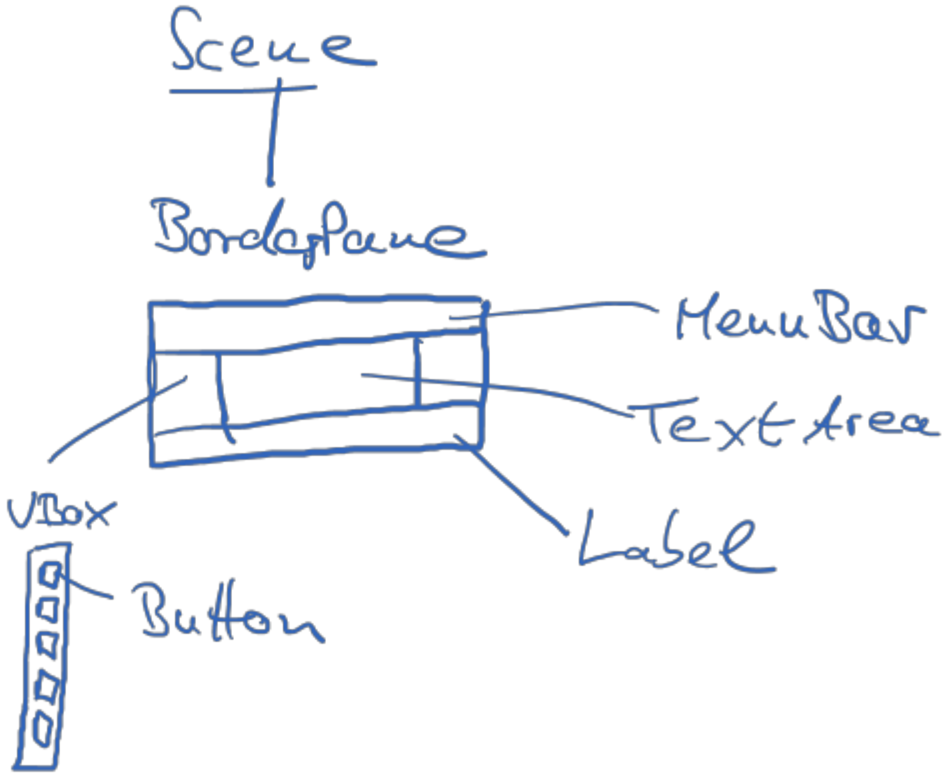


Layout Manager: GridPane

set # rows and columns



Building a Scene Graph



JavaFX Event Handling

What is an Event?

- A Java object that represents a user interaction with the GUI and contains data about the interaction

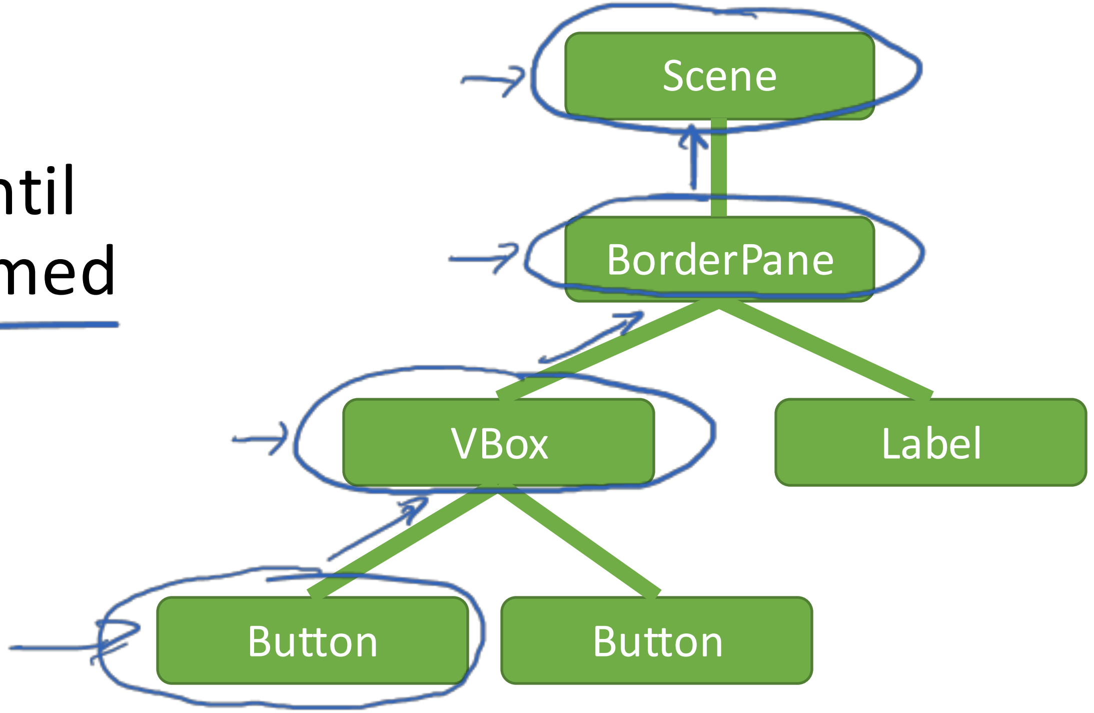
- A subtype of *javafx.event.Event*

  (examples: MouseEvent, KeyEvent, ActionEvent)

Events in the Scene Graph

- Event starts at the control that the user interacts with (the target)

- It then moves up the scene graph until it reaches the root node or is consumed



Event Handlers

- Objects with a method containing code to react to an event

- JavaFX requires this object to be of the *javafx.event.EventHandler* interface type

- Objects of the type *EventHandler* can be registered with a node in the scene graph

The *EventHandler* Interface

```
public interface EventHandler<T extends Event> {

    public void handle (T event);

}
```

Registering EventHandlers

- Scene graph nodes have method

```
.addEventHandler(EventType<T> eventType,
                 EventHandler<? super T> eventHandler)
```

- Examples:
  - .addEventHandler(KeyEvent.KEY_TYPED, EventHandler<KeyEvent> handler)
  - .addEventHandler(MouseEvent.MOUSE_CLICKED, EventHandler<MouseEvent> handler)
  - .addEventHandler(ActionEvent.ACTION, EventHandler<ActionEvent> handler)

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.control.Label;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Polygon;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.geometry.Pos;
import javafx.geometry.Insets;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.event.ActionEvent;
import javafx.application.Platform;

public class JavaFXApp extends Application {

    @Override
    public void start(Stage window) {
        System.out.println("app started...");

        Label label = new Label("Search text:");
        label.addEventHandler(MouseEvent.MOUSE_CLICKED,
                (event) -> System.out.println("Label clicked...") );
        TextField searchText = new TextField();
        searchText.addEventHandler(KeyEvent.KEY_TYPED,
                (event) -> System.out.println("Key typed: " + event.getCharacter() ));
        Button searchButton = new Button("Start Search");
        searchButton.addEventHandler(ActionEvent.ACTION,
                 (event) -> System.out.println("Search Started: " + searchText.getText())
);
        Button closeButton = new Button("Close");
        closeButton.addEventHandler(ActionEvent.ACTION,
                (event) -> Platform.exit() );  → End the program
        //button.setLayoutY(25);

        //Circle circle = new Circle(200, 200, 20);
        //Polygon polygon = new Polygon(100, 150, 180, 90, 35, 80);

        //Group group = new Group(label, button, circle, polygon);

        BorderPane bp = new BorderPane();
        bp.addEventHandler(MouseEvent.MOUSE_CLICKED,
                (event) -> {
            System.out.println("BoderPane has been clicked...");
            event.consume();  → will not propagate up (If other nodes do not have
                 });                the same event handler, you can ignore
        bp.setCenter(searchText);   this line)
        bp.setLeft(label);
        bp.setAlignment(label, Pos.CENTER);
        bp.setMargin(label, new Insets(5, 5, 5, 5));
                          → Gives the padding
        bp.setMargin(searchText, new Insets(5, 5, 5, 0));

        HBox hbox = new HBox(60);
        hbox.getChildren().add(searchButton);
        hbox.getChildren().add(closeButton);
        hbox.setAlignment(Pos.CENTER);
        bp.setBottom(hbox);
        bp.setMargin(hbox, new Insets(5, 5, 5, 5));

        Scene scene = new Scene(bp); //group, 800, 600);
        scene.addEventHandler(MouseEvent.MOUSE_CLICKED,
                (event) -> System.out.println("Scene has been clicked...") );
        window.setScene(scene);

        window.setTitle("JavaFXApp");
        window.show();
    }

    public static void main(String[] args) {
        Application.launch();
    }

}
```